PLEASE NOTE THE OUTPUT IS VERY LONG.
BUT FOR ME, EACH SECTION WAS REQUIRED TO UNDERSTAND THE MECHANICS THAT WAS GOING ON.

********** OUTPUT **********

```
Steps are: [9, 5, 15, 4]
Maximum value of r: 5 has been set using minimum value in set: 4 , which should be in proximity to N(20)
```

Note: Combination.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Welcome to Online IDE!! Happy Coding :)
Steps are: [9, 5, 15, 4]
Maximum value of r: 5 has been set using minimum value in set: 4 , which should be in proximity to N(20)

*****COMBINATIONS*** (WITH REPLACEMENT)** // It can be seen here that 3!/3! Is 1.. But without an r, a selection can not be made.

C^R(n + r) = (n+r-1)! / r!(n-1)!
C^R(4,0) = 3! / 0!(3)!
Combinations: 1
new size: 0
Number of cycles: 1

new size: 0
Number of cycles: 2

new size: 0
Number of cycles: 3

new size: 0
Number of cycles: 4

new size: 0
Number of cycles: 5

new size: 0
Number of cycles: 6

new size: 0
Number of cycles: 7

new size: 0
Number of cycles: 8

new size: 0
Number of cycles: 9

new size: 0
Number of cycles: 10 //it has completed 10 cycles since combinations x 10 used in do while loop as safe measure

*****COMBINATIONS*** (WITH REPLACEMENT)** //There are four different ways to select r=1 n=4
C^R(n + r) = (n+r-1)! / r!(n-1)!
C^R(4,1) = 4! / 1!(3)!
Combinations: 4

random step is: 15 //This is random number
This is performing: 1 of 1(r) //This is number times selection of r will be totalled, even if greater than N (20)
TOTAL HERE: 15 N Value is: 20 //This is running total

new size: 0 //this is indication if criteria has been met
Number of cycles: 1

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 2

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 3

random step is: 4
This is performing: 1 of 1(r)
TOTAL HERE: 4 N Value is: 20
new size: 0
Number of cycles: 4

random step is: 15
This is performing: 1 of 1(r)
TOTAL HERE: 15 N Value is: 20
new size: 0
Number of cycles: 5

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 6

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 7

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 8

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 9

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20

new size: 0
Number of cycles: 10

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 11

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 12

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 13

random step is: 4
This is performing: 1 of 1(r)
TOTAL HERE: 4 N Value is: 20
new size: 0
Number of cycles: 14

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 15

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 16

random step is: 4
This is performing: 1 of 1(r)
TOTAL HERE: 4 N Value is: 20
new size: 0
Number of cycles: 17

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 18

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20

new size: 0
Number of cycles: 19

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 20

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 21

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 22

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 23

random step is: 15
This is performing: 1 of 1(r)
TOTAL HERE: 15 N Value is: 20
new size: 0
Number of cycles: 24

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 25

random step is: 15
This is performing: 1 of 1(r)
TOTAL HERE: 15 N Value is: 20
new size: 0
Number of cycles: 26

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 27

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20

new size: 0
Number of cycles: 28

random step is: 4
This is performing: 1 of 1(r)
TOTAL HERE: 4 N Value is: 20
new size: 0
Number of cycles: 29

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 30

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 31

random step is: 4
This is performing: 1 of 1(r)
TOTAL HERE: 4 N Value is: 20
new size: 0
Number of cycles: 32

random step is: 15
This is performing: 1 of 1(r)
TOTAL HERE: 15 N Value is: 20
new size: 0
Number of cycles: 33

random step is: 5
This is performing: 1 of 1(r)
TOTAL HERE: 5 N Value is: 20
new size: 0
Number of cycles: 34

random step is: 15
This is performing: 1 of 1(r)
TOTAL HERE: 15 N Value is: 20
new size: 0
Number of cycles: 35

random step is: 9
This is performing: 1 of 1(r)
TOTAL HERE: 9 N Value is: 20
new size: 0
Number of cycles: 36

random step is: 4
This is performing: 1 of 1(r)
TOTAL HERE: 4 N Value is: 20

new size: 0
Number of cycles: 37

random step is: 15
This is performing: 1 of 1(r)
TOTAL HERE: 15 N Value is: 20
new size: 0
Number of cycles: 38

random step is: 4
This is performing: 1 of 1(r)
TOTAL HERE: 4 N Value is: 20
new size: 0
Number of cycles: 39

random step is: 15
This is performing: 1 of 1(r)
TOTAL HERE: 15 N Value is: 20
new size: 0
Number of cycles: 40 //it has completed 40 cycles since combinations x 10 used in do while loop as safe measure

*****COMBINATIONS*** (WITH REPLACEMENT)**
C^R(n + r) = (n+r-1)! / r!(n-1)!
C^R(4,2) = 5! / 2!(3)!
Combinations: 10 //this will introduce a cycle count of 100

random step is: 5
This is performing: 1 of 2(r)
TOTAL HERE: 5 N Value is: 20

random step is: 9
This is performing: 2 of 2(r)
TOTAL HERE: 14 N Value is: 20
new size: 0
Number of cycles: 1

random step is: 15
This is performing: 1 of 2(r)
TOTAL HERE: 15 N Value is: 20

random step is: 15
This is performing: 2 of 2(r)
TOTAL HERE: 30 N Value is: 20
new size: 0
Number of cycles: 2

random step is: 15
This is performing: 1 of 2(r)
TOTAL HERE: 15 N Value is: 20

random step is: 5
This is performing: 2 of 2(r)
TOTAL HERE: 20 N Value is: 20 //it can be seen here that total has matched N=20
This is stringjoiner right now: 15,5 //output of string to end user
Count:1
current:0

TOTAL: 20 //confirmation of Total
VALUE OF N: 20 //confirmation of N
COUNT: 1
r: 2
new size: 1 //increase in size of set
This has been added into the set: //confirmation that entry has been added into set (15,5) as above..
Number of cycles: 3

TO KEEP SCREEN OUTPUTS TO MINIMUM, ONLY THOSE WITH TOTAL ARE RETAINED AND LAST ENTRY IN EACH CYCLE FOR C^R(N,R)

IT WILL HAVE REPEAT COMBINATIONS, BUT WILL STORE ONCE….

random step is: 15
This is performing: 1 of 2(r)
TOTAL HERE: 15 N Value is: 20

random step is: 5
This is performing: 2 of 2(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 15,5
Count:1
current:1
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 2
new size: 1
Number of cycles: 26

random step is: 5
This is performing: 2 of 2(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 15,5
Count:1
current:1
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 2
new size: 1
Number of cycles: 31

random step is: 5
This is performing: 1 of 2(r)
TOTAL HERE: 5 N Value is: 20

random step is: 15
This is performing: 2 of 2(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 5,15
Count:1
current:2
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 2
new size: 2
Number of cycles: 86

random step is: 15
This is performing: 1 of 2(r)
TOTAL HERE: 15 N Value is: 20

random step is: 15
This is performing: 2 of 2(r)
TOTAL HERE: 30 N Value is: 20
new size: 2
Number of cycles: 100

5,15 // at end it will show combinations
15,5 // at end it will show combinations

COMBINATIONS (WITH REPLACEMENT)
C^R(n + r) = (n+r-1)! / r!(n-1)!
C^R(4,3) = 6! / 3!(3)!
Combinations: 20

random step is: 5
This is performing: 1 of 3(r)
TOTAL HERE: 5 N Value is: 20

random step is: 15
This is performing: 2 of 3(r)
TOTAL HERE: 20 N Value is: 20 //it can be seen here that even though it has reached- Total 20, sample of size r has not been selected. So there is no attempt to perform 'This has been added into the set'
This is stringjoiner right now: 5,15 //it also shows that the sample right now is too small (2 items)
Count:1
current:2
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 3

random step is: 15
This is performing: 1 of 3(r)
TOTAL HERE: 15 N Value is: 20

random step is: 5
This is performing: 2 of 3(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 15,5
Count:1
current:2
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 3

random step is: 4
This is performing: 3 of 3(r)
TOTAL HERE: 24 N Value is: 20
new size: 2
Number of cycles: 200 //it will have reached its last iteration and still no combination for total = 20 with r=3 since int [] X = new int []{9,5,15,4};

5,15
15,5

*****COMBINATIONS*** (WITH REPLACEMENT)**
C^R(n + r) = (n+r-1)! / r!(n-1)!
C^R(4,4) = 7! / 4!(3)!
Combinations: 35
Based on this, it is expected to hit 20 with sample size =4. In only one possible way {5,5,5,5}..

random step is: 5
This is performing: 1 of 4(r)
TOTAL HERE: 5 N Value is: 20

random step is: 15
This is performing: 2 of 4(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 5,15 ///just to clarify again it will continue since r is only 2 of 4
Count:1
current:2
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 4

random step is: 15
This is performing: 3 of 4(r)
TOTAL HERE: 35 N Value is: 20

random step is: 15
This is performing: 4 of 4(r)
TOTAL HERE: 50 N Value is: 20
new size: 2
Number of cycles: 36

AFTER 130 cycles, it meets {5,5,5,5} out of 350 cycles… The chance of getting all 4's approximately in real world would be 1/4 x 1/4 x 1/4 x 1/4 = 256.. Perhaps do while loop can be reduced.. However I am not knowledged enough on how work out likelihood of it to complete at which cycle… So a 10 fold is continued…
Please note that it will impact the memory of the execution…
So it can be reduced.. but will need to check if combination appears (as highlighted in blue below).
Unfortunately with improvised approach, there is no way to control this scenario..

random step is: 5
This is performing: 1 of 4(r)
TOTAL HERE: 5 N Value is: 20

random step is: 5
This is performing: 2 of 4(r)
TOTAL HERE: 10 N Value is: 20

random step is: 5
This is performing: 3 of 4(r)
TOTAL HERE: 15 N Value is: 20

random step is: 5
This is performing: 4 of 4(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 5,5,5,5

Count:1
current:2
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 4
new size: 3
This has been added into the set:
Number of cycles: 130

random step is: 15
This is performing: 1 of 4(r)
TOTAL HERE: 15 N Value is: 20

random step is: 4
This is performing: 2 of 4(r)
TOTAL HERE: 19 N Value is: 20

random step is: 5
This is performing: 3 of 4(r)
TOTAL HERE: 24 N Value is: 20

random step is: 4
This is performing: 4 of 4(r)
TOTAL HERE: 28 N Value is: 20
new size: 3
Number of cycles: 350

5,15
15,5

5,5,5,5

COMBINATIONS (WITH REPLACEMENT)
C^R(n + r) = (n+r-1)! / r!(n-1)!
C^R(4,5) = 8! / 5!(3)!
Combinations: 56
From here, only solution is {4,4,4,4,4} int [] X = new int []{9,5,15,4};
//It will be interesting to see that it should occur after: 1/4 x 1/4 x 1/4 x 1/4 x 1/4 = 1/1024
There is every chance it is not fulfilled…. Fortuantley it completed on cycle 53

random step is: 4
This is performing: 1 of 5(r)
TOTAL HERE: 4 N Value is: 20

random step is: 4
This is performing: 2 of 5(r)
TOTAL HERE: 8 N Value is: 20

random step is: 4
This is performing: 3 of 5(r)
TOTAL HERE: 12 N Value is: 20

random step is: 4
This is performing: 4 of 5(r)
TOTAL HERE: 16 N Value is: 20

random step is: 4
This is performing: 5 of 5(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 4,4,4,4,4
Count:1

current:3
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 5
new size: 4
This has been added into the set:
Number of cycles: 53

random step is: 4
This is performing: 5 of 5(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 4,4,4,4,4
Count:1
current:4
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 5
new size: 4
Number of cycles: 222

// This is the last cycle and it shows {4,4,4,4,4}

random step is: 15
This is performing: 1 of 5(r)
TOTAL HERE: 15 N Value is: 20

random step is: 5
This is performing: 2 of 5(r)
TOTAL HERE: 20 N Value is: 20
This is stringjoiner right now: 15,5
Count:1
current:4
TOTAL: 20
VALUE OF N: 20
COUNT: 1
r: 5

random step is: 9
This is performing: 3 of 5(r)
TOTAL HERE: 29 N Value is: 20

random step is: 15
This is performing: 4 of 5(r)
TOTAL HERE: 44 N Value is: 20

random step is: 5
This is performing: 5 of 5(r)
TOTAL HERE: 49 N Value is: 20
new size: 4
Number of cycles: 560

4,4,4,4,4

5,15
15,5
5,5,5,5

*****COMBINATIONS*** (WITH REPLACEMENT)**
C^R(n + r) = (n+r-1)! / r!(n-1)!
C^R(4,6) = 9! / 6!(3)!
Combinations: 84
//since 4 is smallest item in r…. it is not possible to reach N=20 with 6 items.
// This is also the last C^R (since it added 1 to maximum R value in the coding)….

random step is: 4
This is performing: 1 of 6(r)
TOTAL HERE: 4 N Value is: 20

random step is: 5
This is performing: 2 of 6(r)
TOTAL HERE: 9 N Value is: 20

random step is: 9
This is performing: 3 of 6(r)
TOTAL HERE: 18 N Value is: 20

random step is: 4
This is performing: 4 of 6(r)
TOTAL HERE: 22 N Value is: 20

random step is: 15
This is performing: 5 of 6(r)
TOTAL HERE: 37 N Value is: 20

random step is: 9
This is performing: 6 of 6(r)
TOTAL HERE: 46 N Value is: 20
new size: 4
Number of cycles: 840

4,4,4,4,4
5,15
15,5
5,5,5,5

** Process exited - Return Code: 0 **

This is the challenge stated in the exercise:

```
Steps are: [1, 2]
Maximum value of r: 5 has been set using minimum value in set: 1 , which should be in proximity to N(4)
```

Exact outcome as documented:

```
2,2
2,1,1
1,1,2
1,2,1
1,1,1,1
```

This is the challenge stated in the exercise:

```
Steps are: [1, 3, 5]
Maximum value of r: 5 has been set using minimum value in set: 1 , which should be in proximity to N(4)
```

```
3,1
1,3
1,1,1,1
```

additional scenario (0 as an integer):

```
Steps are: [4, 0, 0]
Maximum value of r: 2 has been set using minimum value in set: 4 , which should be in proximity to N(4)
```

```
4
0,4
4,0
```

additional scenario (all values larger than N): No combinations..

```
Welcome to Online IDE!! Happy Coding :)
Steps are: [5, 6, 7]
Maximum value of r: 1 has been set using minimum value in set: 5 , which should be in proximity to N(4)
```

*** CODE ***

```
/*
Online Java - IDE, Code Editor, Compiler
Online Java is a quick and easy tool that helps you to build, compile, test your programs
online.
*/
// This has been created to ensure I can utilize any random functions more efficiently.
// It is a creation of the combinations (with replacement) calculator.
// It has used techniques I learnt including recursion and also memoization to speed up execution.
// I will incorporate this into Java applications I created previously..

//TEST CASES
// All done on My Approach
```

```
import java.math.*;
import java.util.*;

class Staircase
{
   static int N = 4;     // this can be changed by end user, size of the staircase


   int[] X;      // this is the array containing the steps
   int p;       // this is random number generated
   int total;   // this is running total of the steps
   int cycles=0;  // this is number of cycles completed for each
   Set<String> s;  // this will contain the combinations for achieving total N

   Random rand = new Random();  // object created of type Random.


   // there is no sense of X getting smaller since its with replacement....
   // need to only add the stringJoiner (converted into string) into the set if the total is equal to N
   // if total goes over, it will continue adding until all cycles for C^R (n,r) has finished


   // Combinations is total of combinations for C^R(n,r)
   // r is the sample size... This can exceed n.

   public Staircase(long combinations, int[] X, int r, Set<String> s)
   {
      this.X=X;
      this.s=s;

      System.out.println("Combinations: " + combinations);

      StringJoiner sj = new StringJoiner(",");  // creating StringJoiner

      int currentSetSize=0;  // holds size of set before entry added
      int newSetSize;      // holds size of set after entry added
      int count;
      int q;             //used in the for loop to iterate through r
      int steps;  // it will hold value of step generated randomly from r

      do
      {
         count=1;
         //sets count back to 1 when it has finished.  This is visual check to ensure that r is not exceeded


        for (q=0; q<r;q++)     // processing up to r only since this is the sample size
        {
            p= rand.nextInt(X.length);  // if length 3, it will give index numbers 0-2... This is suitable
            steps = X[p];  // step value
            System.out.println("\nrandom step is: " + steps);
            sj.add(Integer.toString(X[p]));  //adds the step into StringJoiner
            currentSetSize=s.size();  //current size of set

            total=total+steps;              //keeps running total

            System.out.println("This is performing: " + (q+1) +" of " + r + "(r)");
           //reminding end user value N and running total

           System.out.println("TOTAL HERE:         " + total  +  "      N Value is:  " +  N);

           //if (total==N /*&& count<=r*/)
```

```
            if (total==N)   // running total equals to N (steps in staircase)
            {
               System.out.println("This is stringjoiner right now: " + sj.toString());
               System.out.println("Count:" + count);
               System.out.println("current:" + currentSetSize);
               System.out.println("TOTAL: " + total);
               System.out.println("VALUE OF N: " + N);
               System.out.println("COUNT: " + count);
               System.out.println("r: " + r);

               s.add(sj.toString());  // it will only add it if the conditions are met above (size of r and Total=N)

               count++;
            }

            if (q==r-1)  // if its on the last loop..
            {
               sj = new StringJoiner(",");
               total=0;

            }

         }

         if (total>N)
         {
            total=0;   //resets total
            sj = new StringJoiner(",");  //creates new instance, resets contents...
         }


         newSetSize = s.size();    // new set size
         System.out.println("new size: "+ newSetSize);


         if (newSetSize>currentSetSize)   //if larger, i.e a new combination, it will inform end user..
         {
            System.out.println("This has been added into the set:");
         }

         cycles++;  //increases cycles
         System.out.println("Number of cycles: " + cycles + "\n");

         // can not set this... since it is giving total combinations (with replacement) and not necessarily
      // related to obtaining a total of 6 with each...
      // This can only be placed in speculatively like throwing dice (a fixed time) and getting total....
      //the safest option for the maximum iterations is combinations x 10.

      }while (cycles<combinations*10);

      // this is now getting the String in the set...
      for (String g: s)
      {
         System.out.println(g.toString());
      }

   }
}


public class Combination
{
   public static void main(String[] args)
```

```
{
    System.out.println("Welcome to Online IDE!! Happy Coding :)");


    int originalNumber=0;   // for moment it is declared a 0 to keep next line content...
    int n=originalNumber;
    int r;   // this does not need be in for loop.  user defined
    Set <String> s = new HashSet<>();  // it will contain all combinations...

    int [] X = new int []{5,6,7};  //user defined
    int minimum;  // the smallest value in X. This will assist with generating r value...
    int maxRValue;  //upper limit for r in the loop to process   0<r<=maxRValue+1
    Staircase sc;  //object of type Staircase


    //need to understand that r can be greater than n in replacement.....
    //r is taken from n objects
    // in this example, r can get larger....

    Map <Integer, Long> m;  // this can stay outside loop since its not impacted by size of n and r

    n=X.length;   //length of the array X of steps
    System.out.println("Steps are: " + Arrays.toString(X));

    originalNumber=n;  // this will remain constant during a full iteration..

    m= new HashMap<>();  //new instance is required to start process again...


    //r can be estimated as follows based on:
    //Maximum length combination  is  approximately (N/ minimum value in X)

    minimum=X[0];

    for (int i=0; i<X.length;i++)
        {
          if (X[i]<minimum && X[i]!=0)
          {
             minimum=X[i];
          }
        }

        maxRValue=(Staircase.N)/minimum;

        //informs end user of constraints...
        System.out.println("Maximum value of r: " + (maxRValue+ 1) + " has been set using minimum value in
set: " + minimum + " , which should be in proximity to N(" + Staircase.N + ")");

        //additional 1 added due to potential rounding errors
        for (r=0; r<=maxRValue+1; r++)
        {
          System.out.println("\n***COMBINATIONS*** (WITH REPLACEMENT)");
          System.out.println("C^R(n + r) = " + "(n+r-1)! / r!(n-1)!");
          System.out.println("C^R(" + n+","+r+") = " + (n+r-1)+ "!" + " / " + r+"!"+"("+(n-1)+")!");

          //creates instance of Staircase and main execution of code...
          sc = new Staircase (Combinations (n,r,originalNumber, m), X, r,s);

        }
}

  public static long Combinations (int n, int r, int originalNumber, Map factorialResults)
  {
```

```
      long result=0;
      int denominator1; //denominator split two parts since there are two factorial calculations
      int denominator2; //denominator split two parts since there are two factorial calculations
      int Numerator=n+r-1; // Numerator
      int zero=0;
      long zeroFactorial = 1;

      // if no sample or objects, there are no outcomes...

      if (originalNumber==0 && r==0)
      {
         System.out.println("n and r can not both be equal to zero");
         //System.exit(0);
         return 0;

      }

      //this situation would occur if n is 0 only and r is any positive number accept 0 (if statement above)
      //for instance (C^R (n,r)) = (0,3) 0+3-1 = 2 2<3

      if (originalNumber==0 && originalNumber+r-1<r)
      {
         System.out.println("n+r-1 must be > or = to r");
         //System.exit(0);
         return 0;
      }
      if (Numerator>=1)
      {
         result = ((n+r-1)* (Combinations (n-1, r,originalNumber, factorialResults)));
         // this completes factorial for numerator
         factorialResults.put(Numerator,result); //result stored in the Map
         //factorialResults.put(n-1,result); //result stored in the Map
         //System.out.println("getting result back out numerator: " + (Numerator) + " " + factorialResults.get(n+r-
1));

         if (n==originalNumber) // this will occur once
         {
            denominator1 = r;
            denominator2 = originalNumber-1;
            factorialResults.put(zero,zeroFactorial); //0! is equal to 1

            if (factorialResults.containsKey(denominator1) && factorialResults.containsKey(denominator2))
            {
               long returnValue = result / ((long)factorialResults.get(denominator1) *
(long)factorialResults.get(denominator2));
               return returnValue;
            }
         }

         return result;
      }
      return 1; // it will reach here only when condition not met (Numerator>=1)
   }
}
```